IN THE CLAIMS:

The text of all pending claims, (including withdrawn claims) is set forth below. Cancelled and not entered claims are indicated with claim number and status only. The claims are listed below for the convenience of the Examiner. No amendments have been made. The status of each claim is indicated with one of (original), (currently amended), (cancelled), (withdrawn), (new), (previously presented), or (not entered).

Claims 1-2 (CANCELLED)

3. (PREVIOUSLY PRESENTED) A method according to claim 21, wherein in a call mechanism for a function of an operating system core with which the programs are executed, a call of the accessing process is forwarded to a checking function in which the check is carried out.

4. (PREVIOUSLY PRESENTED) A method according to claim 3, wherein the checking function is integrated into at least one of the address space of the program and the file to be protected as a dynamically integrated file.

5. (PREVIOUSLY PRESENTED) A method according to claim 21,

wherein a call of the accessing process is forwarded to a checking function in which the check takes place, and

wherein the checking function is integrated into an operating system core of an operating system with which the programs are executed.

6. (PREVIOUSLY PRESENTED) A method according to claim 3, wherein the operating system is Windows NT.

7. (PREVIOUSLY PRESENTED) A method according to claim 21, wherein at a predetermined interval of time for each active process that runs along with a program and/or a file to be protected, a check is made to confirm whether the active process is contained in the process file that is assigned to the program and/or the file to be protected, and the process is ended if that is not the case.

8. (PREVIOUSLY PRESENTED) A method according to claim 21, wherein in dependency on a predetermined event for each active process that runs along with a program and/or a file to be protected, a check is made to confirm whether the active process is contained in the process file that is assigned to the program to be protected, and the process is ended if that is not the case.

9. (PREVIOUSLY PRESENTED) A method according to claim 21, wherein a protection program performing the method according to claim 21 is stored encoded and is decoded at the start of the method.

10. (PREVIOUSLY PRESENTED) A method according to claim 9, wherein the programs to be protected are stored encoded and are decoded at the start of the method.

11. (PREVIOUSLY PRESENTED) A method according to claim 9, wherein after the decoding of the protection program, its integrity is checked and the method according to claim 21 is executed only if the integrity of the protection program is assured.

12. (PREVIOUSLY PRESENTED) A method according to claim 11, wherein after the integrity test of the protection program, the integrity of all processes contained in the process files is checked and the method according to claim 21 is executed only if the integrity of all of the processes contained in the process files is assured.

13. (PREVIOUSLY PRESENTED) A method according to claim 12, wherein after the integrity test of the processes, the integrity of the program to be protected is checked and the method according to claim 21 is executed only if the integrity of the program to be protected is assured.

14. (PREVIOUSLY PRESENTED) A method according to claim 13, wherein at least one of the integrity tests takes place through the use of a cryptographic method.

15. (PREVIOUSLY PRESENTED) A method according to claim 14, wherein the cryptographic value is formed through the use of a hash function.

Claims 16-17 (CANCELLED)

18. (PREVIOUSLY PRESENTED) An array according to claim 23, wherein the processor is programmed to execute a call mechanism for a function of an operating system core with which the programs are executed, and a call of the accessing process is forwarded to a checking function to compare the accessing cryptographic value with the cryptographic value stored in the process file.

19. (PREVIOUSLY PRESENTED) An array according to claim 18, wherein the operating system is Windows NT.

20. (CANCELLED)

21. (PREVIOUSLY PRESENTED) A method for protecting several programs from unauthorized access by processes, comprising:

assigning an address space to each program to be protected;

assigning a process file to each program to be protected, where the process file is separate from the program and includes a cryptographic value that uniquely identifies each process that may run in the address space;

determining, during execution of each program to be protected, for each accessing process that attempts to access the address space of the program, whether the accessing process is listed in the process file assigned to the program by

forming an accessing cryptographic value for each accessing process, and

comparing the accessing cryptographic value with the cryptographic value stored in the process file for each accessing process listed in the process file; and

at least one of starting and continuing execution of the accessing process only if said comparing determines a match between the accessing cryptographic value and the cryptographic value stored in the process file for the accessing process.

22. (PREVIOUSLY PRESENTED) A method for protecting several programs from unauthorized access by processes, comprising:

assigning an address space to each program file to be protected;

assigning a process file, separate from the program file, to each program file to be protected, where the process includes at least one cryptographic value, each uniquely identifying a process that may run in the address space;

determining, during execution of each program in each program file to be protected, for each accessing process that attempts to access the address space of the program file, whether the accessing process is listed in the process file assigned to the program file by

forming an accessing cryptographic value for each accessing process, and

comparing the accessing cryptographic value with the cryptographic value stored in the process file for each accessing process listed in the process file; and

at least one of starting and continuing execution of the accessing process only if said comparing determines a match between the accessing cryptographic value and the cryptographic value stored in the process file for the accessing process.

23. (PREVIOUSLY PRESENTED) An array for protecting several programs from unauthorized access by a process, comprising:

a processor programmed to assign an address space and a process file to each program to be protected, where the process file is separate from the program and includes a cryptographic value that uniquely identifies each process that may run in the address space; to determine, during execution of each program to be protected, for each accessing process that attempts to access the address space of the program, whether the accessing process is listed in the process file assigned to the program by forming an accessing cryptographic value for each accessing process and comparing the accessing cryptographic value with the cryptographic value stored in the process file for each accessing process listed in the process file; and to at least one of start and continue execution of the accessing process only if a match is found between the accessing cryptographic value and the cryptographic value stored in the process file for the accessing process.

24. (PREVIOUSLY PRESENTED) An array for protecting several program files from unauthorized access by a process, comprising:

a processor programmed to assign an address space and a process file to each program file to be protected, where the process file is separate from the program file and includes a cryptographic value that uniquely identifies each process that may run in the address space; to determine, during execution of each program in each program file to be protected, for each accessing process that attempts to access the address space of the program file, whether the accessing process is listed in the process file assigned to the program file by forming an accessing cryptographic value for each accessing process and comparing the accessing cryptographic value with the cryptographic value stored in the process file for each accessing

process listed in the process file; and to at least one of start and continue execution of the accessing process only if a match is found between the accessing cryptographic value and the cryptographic value stored in the process file for the accessing process.

25. (PREVIOUSLY PRESENTED) A set of several arrays and a server array connected with each of the several arrays, to protect several programs from unauthorized access by a process, comprising

in each of the several arrays a processor programmed to assign an address space and a process file to each program file to be protected, where the process file is separate from the program file and includes a cryptographic value that uniquely identifies each process that may run in the address space; to determine, during execution of each program in each program file to be protected, for each accessing process that attempts to access the address space of the program file, whether the accessing process is listed in the process file assigned to the program file by forming an accessing cryptographic value for each accessing process and comparing the accessing cryptographic value with the cryptographic value stored in the process file for each accessing process listed in the process file; and to at least one of start and continue execution of the accessing process only if a match is found between the accessing cryptographic value and the cryptographic value stored in the process file for the accessing process.